UNCLASSIFIED

AD NUMBER
AD429018
LIMITATION CHANGES
TO: Approved for public release; distribution is unlimited.
FROM: Distribution authorized to U.S. Gov't. agencies and their contractors; Administrative/Operational Use; JAN 1964. Other requests shall be referred to Defense Advanced Research Projects Agency, ASBD-TIO, 675 North Randolph Street, Arlington, VA 22203-2114.
AUTHORITY
rand ltr, 31 dec 1966

THIS PAGE IS UNCLASSIFIED

429018

ARPA ORDER NO. 189-61

162395

MEMORANDUM
RM-3917-ARPA
JANUARY 1964

(5) 739200

AD No.

DDC FILE COPY

HIGHER-ORDER APPROXIMATIONS TO THE COMPUTATIONAL SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS

Stanley Azen

FEB 13 1964

PREPARED FOR:

ADVANCED RESEARCH PROJECTS AGENCY

The RAND Corporation
SANTA MONICA • CALIFORNIA

429018

2816

NO OTS

16 ARPA ORDER ~~NO.~~ 189-61

5 739 200

14

~~MEMORANDUM~~

Rept. no. RM-3917-ARPA

11 JAN~~UARY 19~~64,

6

HIGHER-ORDER APPROXIMATIONS TO THE COMPUTATIONAL SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS,

10 by Stanley Azen.

15

This research is supported by the Advanced Research Projects Agency under ~~Contract No.~~ SD-79. Any views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of ARPA.

A1

The RAND Corporation

1700 MAIN ST. • SANTA MONICA • CALIFORNIA

PREFACE

Part of the research program of The RAND Corporation consists of basic supporting studies in mathematics. One aspect of this is concerned with the solution of partial differential equations. In this field, the technique of difference approximations has been extremely useful.

In the present Memorandum, the applicability of higher-order difference approximations of an unconventional type to the solution of partial differential equations is investigated.

SUMMARY

In this Memorandum two techniques of approximating the solution to a partial differential equation are investigated. Using the first of these, the solution is approximated at each stage by a higher-order difference algorithm. The second technique is that of storing the function at each stage by a polynomial.

Numerical results were obtained from a FORTRAN program applying these techniques to the equation $u_t = uu_x$.

sub t sub x

ACKNOWLEDGMENT

I would like to express my appreciation to Richard Bellman of The RAND Corporation for suggesting some of the ideas presented in this Memorandum.

HIGHER-ORDER APPROXIMATIONS TO THE COMPUTATIONAL SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS

1. INTRODUCTION

It has been shown in [1] that certain partial differential equations could be solved by an approach which has some advantages over the usual methods of approximation by difference equations. This approach was shown to be superior in the following ways: 1) the solution is re-created at desired points by an approximating polynomial; and 2) the algorithm for approximating the solution exhibits certain desired properties of the actual solution. To illustrate this method it was shown in [2] that the equation

$$u_t = uu_x \,, \quad u(x,0) = g(x) \,, \tag{1.1}$$

which possesses the analytic solution

$$u = g(x + ut) \,, \tag{1.2}$$

where $g(x)$ is here assumed to be an odd function of period two, can be approximated by the algorithm

$$u(x,t+\Delta) = u(x+u(x,t)\Delta,t) \; . \tag{1.3}$$

The variable t was constrained to values $t = 0,\Delta,2\Delta,\ldots,$ but x assumed arbitrary values in the interval [0,1]. The algorithm (1.3) clearly preserves boundedness and non-negativity, and upon expanding in a Taylor series it is seen that the approximation is accurate to $O(\Delta^2)$. In [2] it was suggested that more accurate results could be obtained using a higher-order algorithm. It is the purpose of this paper to investigate the effect of improving the approximation to (1.1) by using an algorithm accurate to $O(\Delta^3)$.

2. HIGHER-ORDER APPROXIMATION

Let the algorithm be given by

$$u(x,t+\Delta) = u(x+u(x+u(x,t)\Delta,t)\Delta,t) \tag{2.1}$$

which also preserves boundedness and non-negativity. Expanding both sides of (2.1) in a Taylor series to $O(\Delta^3)$ and applying the relations

$$u_{tt} = uu_x^2 + uu_{xt} \tag{2.2}$$

$$u_{tx} = u_x^2 + uu_{xx} \tag{2.3}$$

to the result, it can be seen that

$$u_t\Delta + \frac{\Delta^2}{2} u_{tt} + 0(\Delta^3) = uu_x\Delta + \frac{\Delta^2}{2}(2uu_x^2 + u^2u_{xx}) + 0(\Delta^3)$$

holds, and hence, (2.1) is accurate to $0(\Delta^3)$.

Again, let $t = 0,\Delta,2\Delta,\ldots,$ and at each stage of the calculation let $u(x,t)$ be obtained by means of the finite sum

$$u(x,t) \cong \sum_{n=1}^{M} u_n(t) \sin(n\pi x) \; , \tag{2.4}$$

where the coefficients $u_n(t)$ are obtained by the quadrature scheme

$$u_n(t) = 2 \int_0^1 u(x,t) \sin(n\pi x)dx \tag{2.5}$$

$$\cong \frac{2}{R} \sum_{k=1}^{R-1} u(k/R,t) \sin(n\pi k/R) \; . \tag{2.6}$$

Hence, the values $u(k/R,t)$, $k = 1,2,\ldots,R-1$, store $u(x,t)$ at time t, and by way of (2.1) $u(x,t+\Delta)$ can be obtained.

3. NUMERICAL RESULTS

A FORTRAN program for the IBM 7090 was written to compare results obtained using (2.1) with those using (1.3).

(a) $g(x) = 0.1 \sin \pi x$, $0 \leq x \leq 1$

where $M = R = 10$, $\Delta = 0.1$, $0 \leq t \leq 10$

Execution requires 22 seconds using (1.3) and 27 seconds using (2.1). As shown in [2], using (1.3) gives errors in $u(x,t) \leq .003$ for times t less than three. As shown in the following table, using (2.1) usually gives slightly better results than (1.3) for times t less than three.

x	t	exact	(1.3)	error	(2.1)	error
.1	1	.043598	.043058	.00054	.043580	.00018
.3	2	.100000	.100097	.00010	.100159	.00016
.7	2	.055795	.055416	.00038	.055854	.00006
.9	1	.023732	.023579	.00015	.023736	.00000

For $t \sim 10/\pi$, the time at which the shock occurs, the values of $u(x,t)$ using (1.3) were in error about 3 per cent at $x = 0.2$, 0.4, 0.6, and 0.8. Also, the largest errors are made for small x, since the shock occurs at

$x = 0$. It is surprising to note that (2.1) gives poorer results than (1.3) for time $t = 3.1$ (see following table).

x	t	exact	(1.3)	error	(2.1)	error
.1	3.1	.094349	.099887	.00554	.101926	.00758
.7	3.1	.046784	.047194	.00041	.047843	.00106
.9	3.0	.016137	.016212	.00008	.016456	.00032

(b) $g(x) = 0.1 \sin \pi x,\ 0 \le x \le 1$

where $M = R = 10$, $\Delta = 0.05$, $0 \le t \le 10$

To determine the reason that a higher-order algorithm does not give better results near the shock, the same calculation was done using a smaller stepsize Δ. For times t less than three, the accuracy in u calculated using (1.3) is generally better for a smaller Δ; however, the calculation using (2.1) is inconsistent: sometimes better, other times worse. Finally, using (2.1) and a smaller stepsize does not give better results for $t \sim 10/\pi$.

For both algorithms in question, the value $x_1 = u(x+u(x,t)\Delta,t)$ is calculated by means of the polynomial given by (2.4). However, this polynomial is used a second time in calculating $x_2 = u(x,t+\Delta) = u(x+x_1\Delta,t)$ in the case of the higher-order algorithm. Now, if the polynomial approximation error is sufficiently large, then

this error, when propagated calculating x_2, is further increased. This is the reason for the poor results near the shock in examples (a) and (b), above.

(c) $g(x) = 0.1 \sin \pi x$, $0 \le x \le 1$

where $M = R = 20$, $\Delta = 0.05$, $0 \le t \le 10$

x	t	exact	(1.3)	error	(2.1)	error
.1	1	.043598	.043320	.00028	.043593	.00000
.3	2	.100000	.099997	.00000	.099995	.00000
.7	2	.055795	.055591	.00020	.055795	.00000
.9	1	.023732	.024228	.00050	.023733	.00000
.1	3.1	.094349	.091836	.00251	.091602	.00275
.7	3.1	.046784	.046269	.00051	.046397	.00038
Execution Time			87 sec			123 sec

In example (c), however, a 20th degree polynomial is used. As seen in the above table, better results are now obtained. For times less than three, the $O(\Delta^3)$ algorithm gives results precisely to the fifth place, while for $t = 10/\pi$ large errors (mainly those near $x = 0$) are reduced.

4. CONCLUSION

It appears from the results that in order to reduce the computational error by increasing the order of the

algorithm, it is necessary to make a compatible reduction in error in the calculation of (2.4) and (2.6) by increasing M and R as necessary.

REFERENCES

1. Bellman, R. E., Some Questions Concerning Difference Approximations to Partial Differential Equations, The RAND Corporation, RM-3083-PR, April 1962.

2. Bellman, R. E., R. E. Kalaba, and B. Kotkin, On a New Approach to the Computational Solution of Partial Differential Equations, The RAND Corporation, RM-3133-PR, May 1962.

UNCLASSIFIED

UNCLASSIFIED